Copyright ©2020 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

A. Artuñedo, G. Corrales, J. Villagra, and J. Godoy, “Machine learning based motion planning approach for intelligent vehicles,” in 2020 IEEE Intelligent Vehicles Symposium, 2020.

Machine learning based motion planning approach for intelligent vehicles

Antonio Artuñedo, Gabriel Corrales, Jorge Villagra and Jorge Godoy

***Abstract*— The complexity to handle complex situations in automated driving requires increasing computational resources. In this work, we propose a machine learning approach for motion planning aiming at optimizing the set of path candidates to be evaluated in accordance with the driving context. Thus, the computation cost of the whole motion planning strategy can be reduced while generating safe and comfortable trajectories when required. The proposed strategy has been implemented in a real experimental platform and validated in different operating environments, successfully providing high quality trajectories in a small time frame.**

I. INTRODUCTION

Intelligent vehicles must be able to plan their behaviour in a large variety of environments. To safely handle complex situations, autonomous driving requires methods to understand unpredictable situations and react within a short time frame [14, 9]. Furthermore, the complexity of decision-making algorithms is increasing, resulting in computationally intensive tasks [12]. Taking into account the limited computational resources available in the on-board computers of a vehicle, that are shared among different tasks (e.g. perception, localization, control, etc.), the reduction on the computational resources needed for motion planning becomes critical.

In the literature coexist different motion planning techniques. Some of them, based on model predictive control, focus on computing the optimal solution for a planning problem under a set of constraints. These approaches are typically applied to a limited number of driving scenarios, with small prediction horizons and small operating frequencies due to the high computational resources needed [13]. In contrast, other techniques allow a better and more generic exploration of the search space relying on the evaluation of path candidates. However, these techniques also require a lot of computing resources. In this regard, the proposed strategy aims at adapting the size of the path candidates set to the driving context, instead of using a fixed amount of path candidates to be evaluated at each planning stage. Thus, the computation time of the whole motion planning strategy can be reduced.

*This work has been partially funded by the Spanish Ministry of Science, Innovation and Universities with National Project COGDRIVE (DPI2017-86915-C3-1-R), the Community of Madrid through SEGVAUTO 4.0-CM (S2018-EMT-4362) Programme, and by the European Commission through the Project PRYSTINE (ECSEL-783190-2).

Antonio Artuñedo, Gabriel Corrales, Jorge Godoy and Jorge Villagra are with the Centre for Automation and Robotics (CSIC-UPM), Ctra. M300 Campo Real, Km 0,200, Arganda del Rey - 28500 Madrid, Spain. {antonio.artunedo, jorge.villagra, jorge.godoy}@csic.es, lg.corrales@alumnos.upm.es

The approach presented in this work is based on the motion planning strategy proposed in [3], which is integrated in the architecture proposed in [1]. This architecture includes both global and local planning features. In this context, the present work is focused on improving the local planner capabilities, which determines a safe and comfortable trajectory when required. The most suitable path is selected among a set of alternatives that considers the kinematic constraints of the vehicle.

This remainder of the paper is organized as follows: Section II introduces a literature review of the related work. The planning architecture in which this work is framed is introduced in section III. In section IV, the proposed approach for candidates generation using machine learning is presented. In section V the results of the experiments carried out are shown. Finally, the conclusions and future work are discussed in section VI.

II. RELATED WORK

Different artificial intelligence techniques are used in the motion planning literature for autonomous vehicles. In [5], it is proposed to use image encoders based on deep learning that extract features from the visual information of the scene to directly learn optimal control actions regarding steering, throttling and braking.

Some approaches apply reinforcement learning to motion planning processes that are difficult to scale in high dimensions, for example, those related to spatial sampling of configurations or iterative enhancement of an initial trajectory [16]. Therefore, the high dimensionality of the problem is reduced and consequently the search for a solution is facilitated.

Some recent works focuses on learning by imitation using end-to-end approaches. In [17], optimal actions for the upcoming sampling time are predicted by applying a trajectory-planning method based on deep reinforcement learning.

In [8] a deep neural network is used to directly choose a trajectory in a finite prediction horizon. This approach use human driving information together with perception data to apply convolutional neural networks. In comparison to deep reinforcement learning methods, this approach is used to estimate a sequence of optimal states that can be used for motion control [15].

Inverse reinforcement learning techniques are applied in [10]. In this case, manoeuvre demonstrations collected from human drivers are used to extract human driving styles. The authors propose to use learning by demonstration to

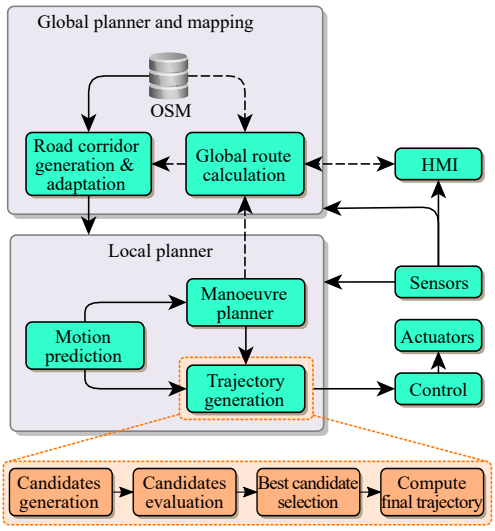


Fig. 1: Functional components of the architecture

capture important properties of a driving model and then reproduce it to generate trajectories.

As reviewed above, most of the approaches found in the literature focus on deep learning applications with strong influence on the decision-making architecture of the vehicle. Nevertheless, rather than raising the problem from an end-to-end approach, in this work we focus on an artificial neural network strategy oriented to increase the performance of an already existing motion planning approach in terms of both computation time and quality of the result.

III. MOTION PLANNING ARCHITECTURE INTRODUCTION

The proposed strategy for motion planning runs within the functional architecture shown in Fig. 1. As can be observed, both global and local planning functionalities are included.

On the one hand, based on a destination coming from the human-machine interface (HMI) and OpenStreepMap data (OSM), the global planner is able to obtain a global route represented by a list of nodes that are later used to compute a road corridor [7].

On the other hand, the local planner block is in charge of computing the final trajectories that the control module will use to generate the final control actions. The motion planning strategy uses the path candidates generation method presented in [3], which comprises the following stages:

1) **Candidates generation**: At this first stage, the motion planning solver defines the search space to explore depending on the planning mode that has been set.
2) **Candidates evaluation**: At this stage all the path candidates are evaluated by checking their validity and calculating their costs based on previously defined cost functions.
3) **Candidate selection**: Among the valid evaluated candidates, the one that minimises a cost value is selected.
4) **Final trajectory calculation**: Once the best candidate is chosen, the speed profile is calculated taking into account the maximum speed and predefined accelerations bounds to ensure comfort inside the vehicle.

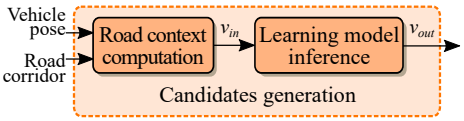


Fig. 2: Proposed method for candidates generation

The main goal of the local planner is to provide a feasible trajectory within the previously generated road corridor. The local planner has been designed to generate smooth trajectories that guarantee a trade-off between comfort and safety inside the vehicle. Moreover, the trajectory must be computed in a reasonable amount of time so that collisions in highly dynamic situations can be avoided.

IV. APPROACH FOR CANDIDATES GENERATION USING MACHINE LEARNING

As stated above, the candidates evaluation is the most computationally expensive stage of this motion planning strategy, since a large number of path candidates have to be evaluated at every time step. However, in some contexts, a significant number of the evaluated candidates are invalid. Hereinafter, we refer to the road section on which the trajectory planning is being carried out as road context.

With the goal of reducing the computational cost of the candidates evaluation stage, a machine learning algorithm is proposed. It will provide an adapted set of candidates to be evaluated, based on the current driving context and the vehicle pose (see Fig. 2). As a result, either the available search space can be more properly assessed in each context – when compared with [3]–, or a reduction of the computation time for the same search space can be achieved.

The following subsections describe in detail the proposed mechanism for candidates set adaptation.

A. Problem statement

The path planning approach relies on the strategy detailed in [3]. This method extracts a set of reference points from the centreline of given road corridor. These reference points are computed using a modified version of the Douglas-Peucker algorithm [6] that states a maximum distance between two consecutive points. Thus, a higher number of reference points is achieved in curved driving contexts.

In the candidates generation, a fixed number of reference points ahead the vehicle (n_{rp}) are used to generate a set of path candidates using quintic Bézier curves. This geometric primitive allows to impose the position (x_i, y_i), orientation θ_i and curvature κ_i at the extreme curve points but also two additional degrees of freedom are still available, which are used to generate a large variety curves with the same initial ($p_0 = [x_0, y_0, \theta_0, \kappa_0]$) and final ($p_f = [x_f, y_f, \theta_f, \kappa_f]$) poses.

In order to generate a set of curves with the same orientation at their extremes, the length of the initial and final velocity vectors ($\vec{\mathbf{t}_0}$ and $\vec{\mathbf{t}_f}$) is varied. Firstly, to make independent the modules of the tangent vectors from each

different curve cases (where the distance between extreme points is not constant), both lengths are normalised with respect to the distance between both curve extremes (d_{0f}). Then, a set of n_t points is generated between the interval $[m_t^{min}, m_t^{max}]$, where m_t^{min} and m_t^{max} are the minimum and maximum normalised lengths of the tangent vectors, respectively. Finally the length of the tangent vector is calculated as follows:

$$|\vec{\mathbf{t}}_0^n| = |\vec{\mathbf{t}}_f^n| = m_{tn} \cdot d_{0f} \quad \forall m_{tn} \in [m_t^{min}, m_t^{max}] \\ n = 1, \ldots, N_t \tag{1}$$

Let $P_0 = [x_0, y_0]$ and $P_f = [x_f, y_f]$ be the position of the initial and final poses, respectively, which in turn are the first and final control points of a Bézier segment. Moreover, $\vec{\mathbf{a}}_n = |\vec{\mathbf{t}_n}|' \cdot \vec{\mathbf{t}_n^u} + \kappa_n |\vec{\mathbf{t}_n}|^2 \cdot \vec{\mathbf{n}_n^u}$ is the acceleration vector, where $\vec{\mathbf{t}_n^u}$, $\vec{\mathbf{n}_n^u}$ and κ_n are the unit tangent and normal vectors and curvature at point P_n, respectively. Then, the position of the four intermediate control points of each Bézier segment can be expressed as a function of these vectors, $\mathbf{P}_m(\vec{\mathbf{t}}_0, \vec{\mathbf{t}}_f, \vec{\mathbf{a}}_0, \vec{\mathbf{a}}_f)$, where $m \in \mathbb{N} : m \in [1, 4]$. In this work, the tangential component of the acceleration vector is considered unitary, while the normal one is computed based on the current vehicle curvature (κ_0) for $\vec{\mathbf{a}}_0$ and is set to 0 for $\vec{\mathbf{a}}_f$. On the basis of the above, a set of quintic Bézier curves can be generated by imposing all combinations of velocity and acceleration vectors at both extremes.

After the validity of the candidate is checked (the path is inside the road corridor and the maximum curvature of the path is lower than the vehicle feasible curvature), the following cost function is computed to evaluate the quality of each path candidate:

$$J_p = \frac{1}{w_{L_p} L_p} \int_{s_0}^{s_f} \dot{\kappa}(s)^2 + w_{\ddot{\kappa}} \ddot{\kappa}(s)^2 \, ds \tag{2}$$

where the length of the path L_p is used to normalize its result. The parameter w_{L_p} is used to weight the length with the curve smoothness, and $w_{\ddot{\kappa}}$ is used to weight the first and second derivatives of the curvature. For further details about the cost function refer to [3, 2].

The fixed parameterization of the original motion planning algorithm is the following: $n_{rp} = 15$, $n_t = 10$, $m_t^{min} = 0.3$, $m_t^{max} = 1.7$, and $n_s = 0.14$, where n_s is the discretization step of the evaluation area ($n_s = \frac{m_t^{max} - m_t^{min}}{n_t}$).

As can be noticed, the geometry of the path candidates generated using this method strongly depends on the values adopted for defining the evaluation area (m_0^{min}, m_0^{max}, m_f^{min} and m_f^{max}). To visualize how the cost of the path candidates are distributed in a given driving context, Fig. 3 represents them with respect to the values of m_0 and m_f. In this figure, a logarithmic scale colour is used to visualize the cost value of each candidate considering (2), while non-valid candidates are coloured in dark blue.

Once the way in which candidates are generated has been established, the objective is to obtain the evaluation area that provides a high percentage of valid candidates around a predicted optimal candidate for each driving situation.

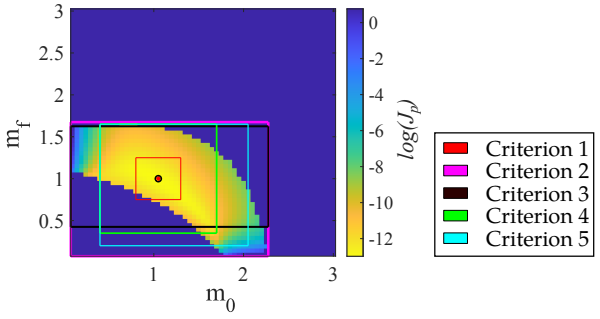


Fig. 3: Distribution of candidates in a given driving context based on m_0 and m_f values. The evaluation areas for the considered criteria are defined by the coloured rectangles

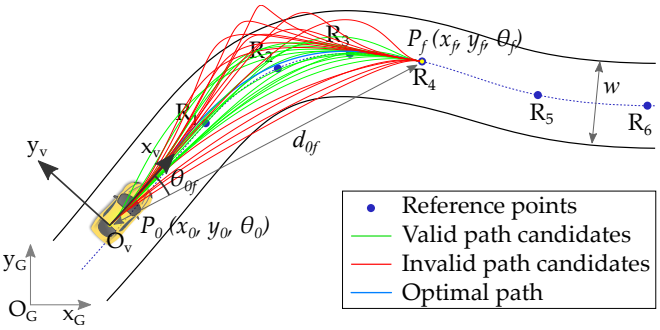


Fig. 4: Driving context example

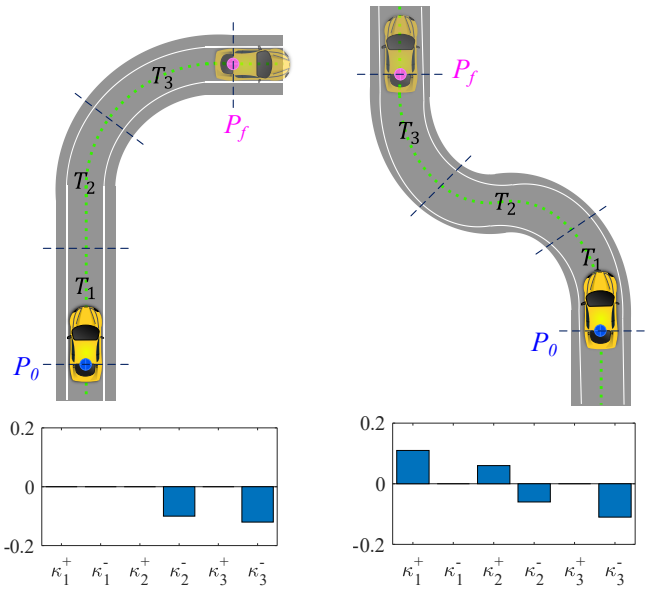
B. Dataset generation

Since the problem will be addressed by using a learning model to predict parameter values, this subsection defines the content of the dataset that is needed for the neural network training and testing stages.

The process to create the dataset can be divided into two main stages: (i) the acquisition of data representative of the performance of the original algorithm for candidate generation, and (ii) the post-processing of the acquired data.

1) Data acquisition: The original path generator has been used for data acquisition along a large set of road corridors. In order to extract a range of data that is able to cover a sufficiently large search space, the path generator has been parameterized with the following values: $m_0^{min} = m_f^{min} = 0.1$, $m_0^{max} = m_f^{max} = 3$ and $n_s = 0.05$. Note that the value of n_s used in the original algorithm is 0.14. It is worth stressing that using these values significantly more candidates are produced in comparison with the original algorithm.

Hereinafter, we refer to the road section on which the trajectory planning is being carried out as road context. The road context is one of the main factors that determines the shape of the admissible paths in a given driving scene. As a result, the dataset must contain representative information of the driving context features. It is considered to include the road corridor width w, the polar coordinates of the reference point being evaluated (d_{0f},θ_{0f}) with respect to the vehicle reference system (O_v), and the centreline curvature (κ) in the section of the road corridor comprised between the vehicle



(a) Right curve at the end sections

(b) Consecutive left and right curves

Fig. 5: Example of road contexts characterization.

position and a given reference point. Besides, the variation between the initial and final heading is also included ($\Delta_\theta = \theta_f - \theta_0$). Fig. 4 shows an schematic example of a driving context where the variables involved in the generated dataset are annotated.

The road corridors used for data generation comprise approximately 25 km of urban and interurban real roads placed in the surroundings of the Centre for Automation and Robotics located at Arganda del Rey, Spain. With this amount of road data it is assumed that a wide range of driving scenarios is sufficiently well represented. To generate the dataset, a large set of approximately 375000 driving contexts were evaluated by imposing the vehicle pose to the centreline of each road corridor at points with a gap of 1 m between two consecutive points. Note that for the same vehicle pose, 15 reference points ($n_{rp} = 15$) ahead the vehicle are evaluated, thus generating 15 new contexts with the same vehicle pose.

2) Data post-processing: This subsection focuses on determining the learning model inputs and outputs as well as defining different criteria for computing the evaluation area.

Together with the lane width w, the curvature along the road (κ) is a representative magnitude of the road geometry. However, curvature along a given road section cannot be directly used as an input to the learning model since it is a continuous value and a finite number of inputs is required. Nevertheless, a description of the curvature can be approximated by the integral of positive and negative curvatures individually. Thus, two different values would represent how sharp a certain section of the road corridor is, to the right and to the left, respectively.

Using just these two values, cases in which the road is curved to both right and left with the same strength can be distinguished (as the one depicted in in Fig. 5b). However, this simple approximation could lead to misrepresentations of the actual road geometry since these two values do not contain information about where the right and left curves are placed within the road section being analysed. To overcome this issue, the curvature along each road corridor section being evaluated is split in various sections with the same length. The number of splits has been set to 3 considering a maximum planning distance of 60 m and assuming insignificant changes in road geometry on a 20 m road section. Thus, the road context is represented in a general manner, allowing to distinguish among the expected driving contexts. Let now consider the road section in Fig. 5a. In this case, the accumulated values of curvature at sections T_2 and T_3 reflects the right curve at the final stretch of road.

Summarizing, the resulting input vector to the learning model is the following:

$$v_{in} = [d_{0f}, \theta_{0f}, \Delta_\theta, w, \kappa_1^+, \kappa_1^-, \kappa_2^+, \kappa_2^-, \kappa_3^+, \kappa_3^-]^T \quad (3)$$

where κ_n^+ and κ_n^- are respectively the positive (left turn) and negative (right turn) integral values of curvature in section n ($n \in \mathbb{N} : n \in [1, 3]$).

With regard to the outputs, note that the predicted values represent a rectangular area over the m_0/m_f graph (see Fig. 3) as follows:

$$v_{out} = [m_0^{min}, m_0^{max}, m_f^{min}, m_f^{max}]^T \quad (4)$$

Given the variability of the distribution of the valid candidates on the m_0/m_f graph, it is worth mentioning the impossibility to state a unique and clear criterion to compute the best rectangular area that is universally adaptable for all possible contexts. Nevertheless, in this work we compare 5 different criteria that aim to obtain a representative area in the m_0/m_f graph that contains the largest amount of valid candidates. Note that in the original algorithm the same interval for the normalized magnitude of the initial and final tangent vector is used, whereas in this work we propose to use different intervals.

All compared criteria are described below:

- **Criterion 1**: This first criterion consists of finding the m_0 and m_f values of the optimal candidate (m_0^{opt}, and m_f^{opt}) using (2). Then, a fixed margin m_{mg} is applied to the optimal point to obtain a square area, resulting $m_0^{min} = m_0^{opt} - m_{mg}$, $m_0^{max} = m_0^{opt} + m_{mg}$, $m_f^{min} = m_f^{opt} - m_{mg}$ and $m_f^{max} = m_f^{opt} + m_{mg}$, where m_{mg} has been set to 0.25.
- **Criterion 2**: This criterion corresponds to a search space that contains all valid candidates, which results in a rectangular area with varying width and height.
- **Criterion 3**: This criterion is built on top the rectangle obtained with criterion 2. The cropping is done here in an iterative manner, reducing one side of the rectangle at a time until a minimum of 60% of valid candidates are reached within the search space. To determine which side is cropped in each iteration, it is verified in which of the four sides there is a greater number of invalid candidates.

- **Criterion 4**: In this case, the coordinates of the optimal candidate are identified and a rectangular region is expanded in steps of 0.05 from this point until a minimum 60% of valid candidates exist within this region. The expansion of the region is carried out equally on all four sides, so that square shapes are obtained.
- **Criterion 5**: This criterion is a variation of the previous one, aiming at covering a larger area. However, in this case the expansion order is determined by the rectangle side in which there is a greater number of invalid candidates (as in criterion 3).

The computation of the resulting evaluation areas from the different criteria introduced above are included in the dataset that will be used for training. An example of the resulting areas is shown in Fig. 3, where each coloured rectangle depicts the resulting evaluation area of each specific criterion.

C. Learning model setup

Given that the outputs of the learning model are the values that define the evaluation area (m_0^{min}, m_0^{max}, m_f^{min} and m_f^{max}), the stated problem can be modeled as a nonlinear regression. The machine learning technique used in this work is the multi-layer perceptron (MLP) [11], which gives in principle good results for this type of problem, while offering acceptable inference time when the size of the neural network is small.

Before the training stage, the normalization of input data is required to avoid the saturation of activation functions. The normalization criterion used is to achieve zero mean and unit standard deviation at each coordinate separately, as follows:

$$x_{nj} = \frac{x_j - \overline{x}}{\sigma_x} \tag{5}$$

where x_{nj} is the normalized value of x_j, and $\overline{x}$ and σ_x are the mean and standard deviation of all samples of variable x, respectively.

Before training, the dataset has been randomly ordered and split in two parts: the training set, containing the 80% of total data and the testing set, containing the remaining 20%.

In order to minimize inference time while obtaining accurate predictions, different configurations of the MLP model have been tested. These configurations comprise 1 to 3 hidden layers and 10 to 80 neurons per layer. As a starting point, an equal number of training epochs in all the models were considered, so that the different MLP configurations can be compared among them. Using 1000 epochs for training, the model that preliminarily offers the best results comprises three hidden layers, whose size is 80, 60, and 10 neurons, respectively. With regard to activation functions, sigmoid function is used in the hidden layers and linear function is applied in the output layer.

For the comparison among the different criteria detailed above, the training has been performed with the Scaled Conjugate Gradient algorithm included in Matlab toolbox for deep learning. This method provides a good balance between training time and test error. Fig. 6 shows the mean squared error (MSE) with respect to training epochs. In this figure it can be observed that after approximately 500 training epochs the difference in the decrease of MSE of training and test data becomes noticeable. In addition, it is observed that from 1000 epochs onwards the test error tends to grow. On the basis of this analysis, a training limit of 700 epochs is adopted. Although this graph has been obtained using the data of criterion 1, a similar trend is observed with all the other criteria.

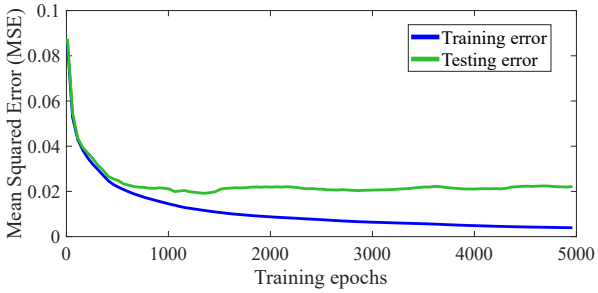


Fig. 6: Mean Squared Error of the model outputs with respect to training epochs using the criterion 1

TABLE I: Simulation results of best MLP structures for each criterion

Criterion	1	2	3	4	5
MLP structure	40-40	60-20-20	60-20	50-30	50-10-10
MSE	0.0333	0.0629	0.0813	0.0644	0.0794
Inference time (ms)	5.57	6.24	5.96	5.60	6.16

Different models were trained for each of the criteria listed above. To assess the generalization of the trained models against different input data, a k-fold cross-validation with 5 iterations has been conducted for each of them.

The summary of the best models obtained with the cross-validation process is summarised in Table I. The results obtained in this comparison show small output errors when the size of the network exceeds a certain minimum size (layers and neurons) depending on each criterion. Bearing this in mind, and being computing time critical for the proposed problem, the model that provides the shortest inference time while offering small output error has been selected.

V. EXPERIMENTAL RESULTS & DISCUSSION

This section focuses on testing and validation of the proposed machine learning approach for motion planning. Firstly, simulation tests were carried out to compare the results obtained with different criteria to build the evaluation area. Moreover, the proposed algorithm was implemented and integrated on an experimental vehicle and several trials on real environments were performed.

A. Criteria comparison in simulation

The different criteria considered to determine the evaluation area in the m_0/m_f graph were tested in different simulated scenarios extracted from real roads of the Centre for Automation and Robotics surroundings, in Arganda del

Rey, Spain. The following two scenarios were chosen to highlight the key aspects of the prediction models in curved road sections where the percentage of valid solutions is typically lower than in scenarios with straight roads:

1) Scenario 1: Continuously curved road: A 75 m curved section of an interurban road is analysed. The road section and the curvature integral values are shown in Fig. 7a and Fig. 7b, respectively. Both the expected values (solid lines) and the resulting values (dashed lines) obtained from the machine learning approach for each criterion are depicted over the m_0/m_f graph the shown in Fig. 7d. In this figure the predicted evaluation area can be visually compared to the ideal output for each criterion. Furthermore, Fig. 7c shows the computed errors between the prediction and the expected values of the evaluation area bounds (m_0^{min}, m_0^{max}, m_f^{min} and m_f^{max}).

Note that most of the predictions have small errors in this scenario except in the case of criterion 3 (black), which presents a big difference with respect to the expected value. Despite the fact that both criterion 1 and 2 have small prediction errors, using criterion 1 (red) results in a much greater percentage of valid candidates and a smaller evaluation region, leading to a shorter computation time of the motion planning evaluation stage.

2) Scenario 2: End-curved road: A 45 m section of an urban road is analysed. The scenario presents a sharp left curve at the end of the road corridor, as depicted in Fig. 8a. In other words, the negative cumulative value of curvature of its section 3 is high, as can be observed in Fig. 8b.

The resulting prediction errors are shown in Fig. 8c. In this case, it can be noticed that the shape of the region including the valid candidates over the m_0/m_f graph is significantly different from the one obtained in the previous scenario. It can be seen that criterion 1 (red) is the one with the smallest prediction error. while offering a limited amount of candidates to be evaluated in the surroundings of the optimal one.

B. Trials in real scenarios

Given the good results obtained in simulation in terms of computation time and quality of the outputs, the proposed strategy for motion planning using machine learning has been tested and validate on real scenarios.

The vehicle used in the trials is a Citroën DS3 which includes hardware modifications for the automated control of throttle, brake, gearbox and steering systems (see Fig. 9). The localization relies on a RTK DGPS receiver. The vehicle also includes an on-board computer with an Intel Core i7-3610QE and 8GB RAM.

In view of the results obtained in the criteria comparison above, the selected approach to be used in the implementation for real testing is criterion 1, since it presents the most stable results concerning the size of the output evaluation area while offering smalls prediction errors. As can be seen in Table I, the best model for criterion 1 comprises 2 hidden layers and 40 nodes in each layer.

TABLE II: Timing results of real trials

Trial	1		2	
Total planning time of original algorithm	58.66		50.17	
Discretization	$n_t = 4$	$n_t = 7$	$n_t = 4$	$n_t = 7$
Inputs computation time	0.1047	0.0956	0.0741	0.0842
Inference time	0.1317	0.1241	0.0979	0.1071
Total planning time	32.11 (1.83x)	40.38 (1.45x)	31.69 (1.58x)	42.17 (1.19x)

Times are expressed in milliseconds.

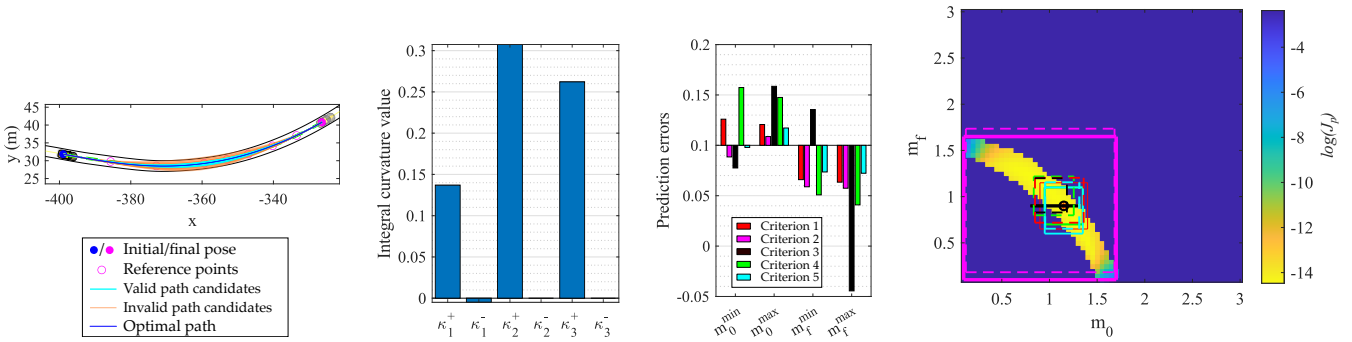
The MLP model for criterion 1 was implemented in C++ using the open source library *mlpack* [4], without using GPU acceleration. Due to the unavailability in *mlpack* library of the training algorithm used in Matlab simulations, the Stochastic Gradient Descent algorithm was used for training the model in the C++ implementation.

Two different tracks were used to test the proposed algorithm in real scenarios (see Fig. 10). In the first scenario, the tests were carried out considering different discretization values for the search space computed by the learning model, that is, n_t has been tested with values 4, and 7. In order to have the original algorithm performance as baseline, note that the discretization step with $n_t = 4$ in the criterion 1 corresponds to a discretization step similar to the one used in the original motion planning algorithm. Indeed, the size of the evaluation area is constant using this criterion ($m_0^{max} - m_0^{min} = m_f^{max} - m_f^{min} = 0.5$, and $n_s = \frac{m_t^{max} - m_t^{min}}{n_t} = 0.125$, while in the original algorithm $n_s = 0.14$).

Table II shows the mean planning time during both tests using the proposed planning strategy with two different discretization steps, as well as using the original planning algorithm. As can be noticed, using a similar discretization step than the original algorithm ($n_t = 4$), in both cases the computation time is significantly reduced with respect to the original (speedup of 1.83x and 1.58x in each scenario, respectively). If a higher discretization step is used to make a deeper exploration of the search space ($n_t = 7$), reductions of approximately 1.45x and 1.58x are respectively obtained.

To analyse the overhead introduced by the learning model in the candidates generation stage of the motion planning strategy, the computation time of the inputs needed for the learning mode and its inference time were also measured during the tests. These results are shown in Table II. As can be seen, both computation times (inputs computation and inference time) represent less than the 1% of the total planning time in all analysed cases. In consequence, the overhead introduced by the learning model approach is negligible.

With regard to the number of valid candidates, while the original algorithm obtains a mean percentage value of 22.9% (trial 1) and 19.4% (trial 2), using the proposed machine learning approach 36.1% and 23.2% are respectively achieved. It shows a noticeable improvement in the generation of valid candidates with respect to the original algorithm since a higher percentage of valid candidates is achieved.



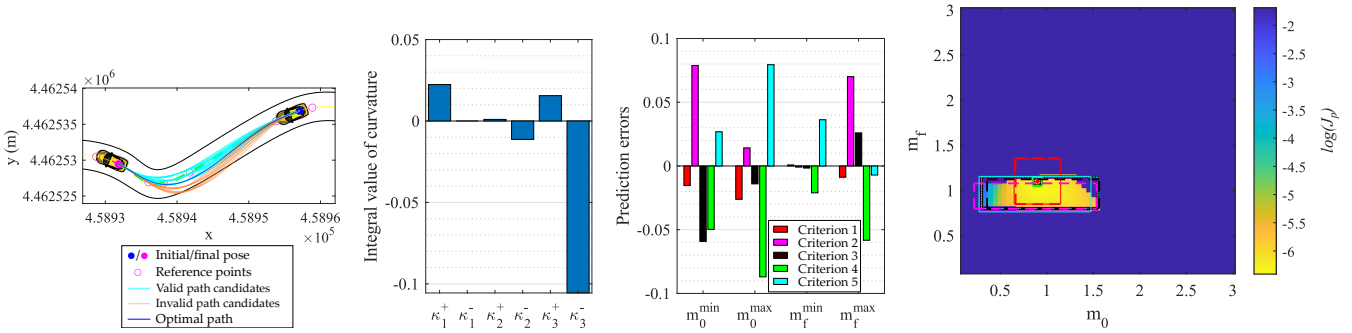
(a) Road section

(b) Road context characterization

(c) Prediction errors

(d) m_0/m_f graph. Expected and output evaluation areas for the considered criteria

Fig. 7: Results in scenario 1: Continuously curved road



(a) Road section

(b) Road context characterization

(c) Prediction errors

(d) m_0/m_f graph. Expected and output evaluation areas for the considered criteria

Fig. 8: Results in scenario 2: End-curved road

Fig. 9: Experimental platform

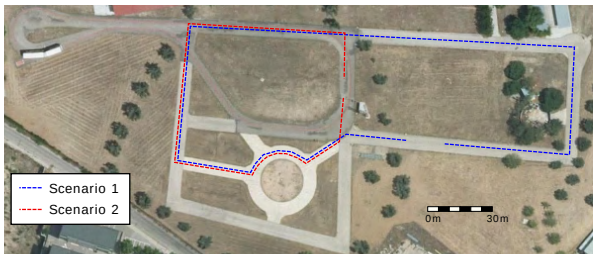


Fig. 10: Trials scenarios

Furthermore, a profitable saving of processing time is also accomplished since a lower amount of candidates must be evaluated.

VI. CONCLUSIONS

In this work, the design, implementation and validation of a motion planning approach for intelligent vehicles using machine learning is proposed.

A first identification of variables that can be used to characterise a driving context has been conducted. This characterisation allows to produce the needed data about the driving context in order to generate a large dataset for neural network training purposes. A comparison among five different criteria has been carried out to determine the most appropriate method for computing the desired output. Moreover, different setups of the MLP model have been compared in simulation for each of the proposed criteria. This comparison allowed to choose the best approach for a later implementation stage.

The proposed startegy has been tested in different real scenarios on an experimental platform. The results obtained shows a clear reduction on the computation time of the

planning algorithm, while preserving the quality of the planned trajectory.

From the promising results obtained in this work, future activities will be focused on extending the dataset in different ways in order to consider a wider range of driving situations. Furthermore, given the good inference time obtained in this work, the input vector can be extended to improve the context representation to increase the performance of the learning model.

REFERENCES

[1] A. Artuñedo, J. Godoy, and J. Villagra. "A decision-making architecture for automated driving without detailed prior maps". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. June 2019, pp. 1645–1652. DOI: 10.1109/IVS.2019.8814070.

[2] A. Artuñedo, J. Godoy, and J. Villagra. "A Primitive Comparison for Traffic-Free Path Planning". In: *IEEE Access* 6 (2018), pp. 28801–28817. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2839884.

[3] A. Artuñedo, J. Villagra, and J. Godoy. "Real-Time Motion Planning Approach for Automated Driving in Urban Environments". In: *IEEE Access* 7 (2019), pp. 180039–180053. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2959432.

[4] Ryan R. Curtin et al. "mlpack 3: a fast, flexible machine learning library". In: *Journal of Open Source Software* 3 (26 2018), p. 726. DOI: 10.21105/joss.00726. URL: https://doi.org/10.21105/joss.00726.

[5] "Deep Reinforcement Learning framework for Autonomous Driving". In: *Electronic Imaging* 2017.19 (2017), pp. 70–76. ISSN: 2470-1173. DOI: doi:10.2352/ISSN.2470-1173.2017.19.AVM-023.

[6] David H. Douglas and Thomas K. Peucker. "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature". In: *Classics in Cartography: Reflections on Influential Articles from Cartographica* 10.2 (Dec. 2011), pp. 15–28. ISSN: 0317-7173. DOI: 10.1002/9780470669488.ch2.

[7] J. Godoy, A. Artuñedo, and J. Villagra. "Self-Generated OSM-Based Driving Corridors". In: *IEEE Access* 7 (2019), pp. 20113–20125. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2897348.

[8] S. M. Grigorescu et al. "NeuroTrajectory: A Neuroevolutionary Approach to Local State Trajectory Learning for Autonomous Vehicles". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019), pp. 3441–3448. ISSN: 2377-3774. DOI: 10.1109/LRA.2019.2926224.

[9] Sorin Grigorescu et al. "A survey of deep learning techniques for autonomous driving". In: *Journal of Field Robotics* (2019).

[10] T. Gu, J. M. Dolan, and J. Lee. "Human-like planning of swerve maneuvers for autonomous vehicles". In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. June 2016, pp. 716–721. DOI: 10.1109/IVS.2016.7535466.

[11] Simon Haykin. *Neural Networks: A Comprehensive Foundation (3rd Edition)*. USA: Prentice-Hall, Inc., 2007. ISBN: 0131471392.

[12] Christos Katrakazas et al. "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions". In: *Transportation Research Part C: Emerging Technologies* 60 (Nov. 2015), pp. 416–442. ISSN: 0968090X. DOI: 10.1016/j.trc.2015.09.011.

[13] M. Nolte et al. "Model predictive control based trajectory generation for autonomous vehicles — An architectural approach". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. June 2017, pp. 798–805. DOI: 10.1109/IVS.2017.7995814.

[14] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and Decision-Making for Autonomous Vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (May 2018), pp. 187–210.

[15] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and Decision-Making for Autonomous Vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (2018), pp. 187–210. DOI: 10.1146/annurev-control-060117-105157.

[16] Changxi You et al. "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning". In: *Robotics and Autonomous Systems* 114 (2019), pp. 1 –18. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2019.01.003.

[17] Lingli Yu et al. "Intelligent Land-Vehicle Model Transfer Trajectory Planning Method Based on Deep Reinforcement Learning". In: *Sensors* 18.9 (2018). ISSN: 1424-8220. DOI: 10.3390/s18092905.